目　录

1、 UNIX 系统部分 4

1.1、 用户的理解 4

1.1.1、 能正确理解 UNIX 系统下的用户概念 4

1.1.2、 能正确掌握增删用户的操作 4

1.1.3、 能正确理解环境变量与用户的关系 4

1.1.4、 能正确理解用户 ID、用户的组、以及用户所具有的权限 4

1.2、 文件系统 4

1.2.1、 能正确理解 UNIX 文件系统的概念 4

1.2.2、 能正确理解和掌握文件的权限以及文件类型 4

1.2.3、 能正确掌握文件的基本操作 4

1.3、 进程管理 5

1.3.1、 能正确了解 UNIX 系统下的进程的概念 5

1.3.2、 能正确了解 UNIX 系统的进程调度 5

1.3.3、 能正确理解父子进程 5

1.3.4、 能正确掌握对进程的基本操作 5

1.4、 常用指令 5

1.4.1、 VI 的操作 5

1.4.2、 FTP 的操作命令 5

1.4.3、 其他操作 5

2、 程序部分 6

2.1、 C 语言部分 6

2.1.1、 文件操作 6

2.1.2、 字符串操作 6

2.1.3、 内存操作 6

2.1.4、 形参和值参 6

2.1.5、 函数定义和返回值 7

2.1.6、 头文件的使用 7

2.1.7、 Base 库的理解 7

2.2、 数据库嵌入式程序部分 7

2.2.1、 能正确掌握编写嵌入式数据库程序 7

2.2.2、 能正确掌握索引 7

2.2.3、 能正确掌握数据库事物 7
2.2.4、 能正确掌握数据库事物 8
2.2.5、 能正确掌握打开/关闭数据库的方法 8
2.2.6、 能正确掌握游标的类型与使用方法 8
2.2.7、 能正确掌握数据库锁的概念 8
2.2.8、 能正确理解 SQLCODE、SQLERRD[2] 8
3、 DBACCESS 操作要求 8
3.1、 基本操作 8
3.2、 能正确使用 FINDERR 检查数据库错误原因 8
4、 其它 8
4.1、 能正确掌握工程文件 MAKEFILE 的使用 9
4.2、 能比较熟练的阅读计算机专业英语 9

1、 UNIX 系统部分

1.1、 用户的理解

1.1.1、 能正确理解 UNIX 系统下的用户概念

能理解 UNIX 系统下的用户和 WINDOWS 系统的用户的区别

Windows 系统功能只限于向超级用户提供，超级用户对系统有自由的支配权。而 unix 操作系统则有权限的限制，某些除超级用户外的用户只有少数的许可权。所以 unix 的安全性更高。

1.1.2、 能正确掌握增删用户的操作

以 SCO OPENSERVER 为例，能在该 UNIX 系统下面进行增加一个用户的操作，删除一个用户的操作。

要在 UNIX 系统中增加新用户需要下列步骤：

（1）在/etc/passwd 文件中新增一行数据，表示给该用户的信息；

（2）为该用户选择工作组，将该用户标识符加在/etc/group 文件内适当位置；

（3）为该用户创建一个家目录（home directory）,将其家目录的拥有者改为该用户所有，组别也改为该用户的组别；

（4）为该用户设置密码。

一、/etc/passwd 文件

/etc/passwd 是一文本文件，用来存放用户信息，每行表示一个用户。

/etc/passwd 每行的格式如下：

user_name:password:uid:gid:comment:home:shell

每行有很多项组成，项与项之间用":"隔开。每项的说明如下：

user_name	用户名
password	登录密码，初始设置时为空
uid	用户识别号(User ID),是一数值，每个用户的识别号不同
gid	工作组识别号，参见/etc/group文件
comment	注释，可以任意字符，一般用来说明用户的身份特征
home	家目录名
shell	该用户缺省shell，一般取值为：/bin/sh、/bin/ksh、/bin/csh

/etc/passwd 的内容举例如下：

cfl:x:201:50:Cao Feilong:/usr/cfl:/bin/sh

abc:x:206:50:abc:/usr/abc:/bin/sh

要增加一新用户，用户名 devos，需要在/etc/passwd 文件末尾增加如下一行：

devos::207:50:Dick Devos:/usr/devos:/bin/sh

二、/etc/group 文件

/etc/group 是一文本文件，用来存放用户工作组信息，每行表示一个组。/etc/group 每行的格式如下：

group_name:password:gid:members_list

每行有四项组成，项与项之间用":"隔开。

group_name	工作组名
password	工作组密码，一般为空
gid	工作组识别号(Group ID)，是一数值，每个组的识别号不同
members_list	该组成员列表，由一个或多个用户名组成，用户名之间用逗号隔开

/etc/group 的内容举例如下：

other::1:root,daemon

group::50:ingres,sybase,cfl,abc

值得说明的是，一个用户可以是多个组的成员。

若要使 devos 用户属于 group 组，还需要在/etc/group 文件中 group 组所在行末加上 devos：

group::50:ingres,sybase,cfl,abc,devos

三、创建家目录

mkdir /usr/devos　　在/usr 目录下创建与用户名同名的目录 devos

chown devos /usr/devos　　设置/usr/devos 的属主为 devos

chgrp group /usr/devos　　设置/usr/devos 的工作组为 group

chmod 755 /usr/devos　　设置存取权限，一般为 755

四、设密码

对用户 devos 建立密码的命令为：

passwd devos

早期的 UNIX 版本，用户密码经加密后存放在/etc/passwd 中的 password 字段。而在 SVR3 以后的版本则借用/etc/shadow 存放用户密码。

6.2 删除用户

删除用户的步骤如下：

（1）删除/etc/passwd 内该用户的信息行

（2）删除/etc/group 内有关该用户的项。

（3）将该用户的家目录删除

1.1.3、能正确理解环境变量与用户的关系

知道一个用户的环境变量在什么地方可以设置、以及如何修改环境变量。

1、Linux 是一个多用户的操作系统。每个用户登录系统后，都会有一个专用的运行环境。通常每个用户默认的环境都是相同的，这个默认环境实际上就是一组环境变量的定义。用户可以对自己的运行环境进行定制，其方法就是修改相应的系统环境变量。

2、常在/etc/profile 文件中修改环境变量，在这里修改的内容是对所有用户起作用的。

以下主要操作将在该文件中进行。

3、如何添加环境变量。

例如添加“NAME=liheng”。

在 profile 文件的最后添加如下内容 export NAME=liheng

变量值 liheng 可以加引号也可以不加，效果一样。

4、在 profile 文件添加或修改的内容需要注销系统才能生效。

5、重复定义变量怎样理解。

经常出现的是对 PATH 变量的定义。

例如：在 peofile 文件默认对 PATH 变量都有设置 PATH=￥￥￥￥￥￥￥，在以后可能在对 PATH 设置，一般都加在 profile 文 件的最后 PATH=……(打个比方)。而系统之中认定的 PATH=……￥￥￥￥￥￥￥￥￥，也就是说相同名字的环境变量，后写入的先起作用。大家一定要注意。

6、特殊字符介绍。

例如在 profile 中有如下内容，通过以下内容说明特殊符号的用法。

export A=/q/jing:aaa/cc/ld

export B=.:/liheng/wang

export A=/cd/cdr:$A

大家注意红色的符号:

: 表示并列含义，例如 A 变量值有多个，用：符号进行分离。

. 表示你操作的当前目录。例如 pap 命令会查找 B 环境变量。

在/home 键入 pap 命令，系统首先在/home 目录下(即当前路径)查找关于 B 的内容，如果没有在/liheng/wang 目录下查找关于 B 的内容。

$ 表示该变量本次定义之前的值，例如$A 代表/q/jing:aaa/cc/ld。也就是说

A=/cd/cdr:/q/jing:aaa/cc/ld

7、使用 env 命令显示所有的环境变量 。在命令提示符下键入 env 就行了。

set 命令显示所有本地定义的 Shell 变量。

8、常见的环境变量

PATH：决定了 shell 将到哪些目录中寻找命令或程序

HOME：当前用户主目录

MAIL：是指当前用户的邮件存放目录。

SHELL：是指当前用户用的是哪种 Shell。

HISTSIZE：是指保存历史命令记录的条数

LOGNAME：是指当前用户的登录名。

HOSTNAME：是指主机的名称，许多应用程序如果要用到主机名的话，通常是从这个环境变量中来取得的。

LANG/LANGUGE：是和语言相关的环境变量，使用多种语言的用户可以修改此环境变量。

PS1：是基本提示符，对于 root 用户是#，对于普通用户是$。

PS2：是附属提示符，默认是“>”。可以通过修改此环境变量来修改当前的命令符，比如下列命令会将提示符修改成字符串“Hello,My NewPrompt :) ”。

PS1=" Hello,My NewPrompt :) "

9、使用修改.bashrc 文件进行环境变量的编辑，只对当前用户有用。

使用修改 /etc/profile 文件进行环境变量的编辑，是对所有用户有用。大家一定要注意区别。

10、profile 文件在系统启动时将被运行。大家可以在里面加入其他命令，但是一定要加正确，不然的话系统会启动不起来的。

1.1.4、能正确理解用户 ID、用户的组、以及用户所具有的权限

了解什么是用户ID口令文件登录项中的用户I D（user ID）是个数值，它向系统标识各个不同的用户。系统管理员

在确定一个用户的登录名的同时，确定其用户I D。用户不能更改其用户I D。通常每个用户有一个唯一的用户 I D。

了解什么是用户的组口令文件登录项也包括用户的组I D（group ID），它也是一个数值。组I D也是由系统管理
员在确定用户登录名时分配的。一般来说，在口令文件中有多个记录项具有相同的组I D。在UNIX下，组被用于将若干用户集合到课题或部门中去。这种机制允许同组的各个成员之间共享资源(例如文件)。4.5节将说明可以设置文件的许可权使组内所有成员都能存取该文件，而组外用户则不能。

了解用户具有哪些权限这两天上网总有人问我UNIX里的文件后面的rwxr-xr--,754是什么意思，知道是代表权限，不知具体是什么意思！正好我的网站文章更新（就是我到各大网站，把好文章拿回来放到我站上。这次100多篇呢）正好124篇，差一篇125，为了凑正我就写了这篇文章，一举两得嘛！:)

好了进入正题，UNIX为了对文件进行保护，UNIX系统中提供了文件存取控制方式。把所有用户划分为三种身份，依次是：文件主（user)、同组用户(group)和其他用户（other).

每种用户对一个文件都拥有读(r)、写(w)和执行(x)的权限。这样就用9个二进制位表示文件的存取方式。例如：

111101100（对应的八进制数表为754）

表示文件主（前三位，111）对该文件的权限有可读、可写、可执行。

同组用户（中间三位，101）对该文件的权限有可读、不可写，可执行。

其他用户（后三位，100）对该文件的权限有可读、不可写、不可执行。

（其中1表示在该位置上有权限，0表示在改位置上无权限。）

用字符表示上述文件方式如下所示：

rwxr-xr--

这9位的顺序是固定的。前三位文件主、中间3位同组用户、后三位其他用户。

（其中“-”的位置表示对应用户不具备相应的权限）

1.2、 文件系统

1.2.1、 能正确理解 UNIX 文件系统的概念

知道 UNIX 文件系统的概念、掌握目录、文件的树型结构

UNIX 操作系统可由多个可以动态安装及拆卸的文件系统组成。UNIX 文件系统主要分为两大类：根文件系统和附加文件系统。

根文件系统（the root file system） 每一个 UNIX 操作系统在其主硬盘上至少含有一个文件系统，它包含构成操作系统的程序和目录，一般由“/”符号来表示。

附加文件系统　除根文件系统外的其它文件系统，如/u 文件系统，AFS 文件系统等。

在 UNIX 中文件共分为四种：

1) 一般文件(ordinary file),分为:
   (1) 文本文件
   (2) 二进制文件
2) 目录文件(directory)
3) 特殊文件(special file)，分为:
   1 块设备文件
   2 字符设备文件
4) 符号链接文件(symbolic links)

文件类型标识　在用"ls -l"命令显示文件目录时的用下列符号表示不同的文件类型：

- 普通文件
d 目录文件
b 块设备文件
c 字符设备文件
p 管道设备
l 符号链接文件

举例：

```
$ ls 朽
-rwxr-xr--  2 wjm newservice 321 Oct 17 09:33 file1
drwxr-xr-x  2 wjm newservice 96  Oct 17 09:40 dir1
```

其中，第一列的"-"表示 file1 是普通文件，"d"表示 dir1 为目录文件。

UNIX 树型目录结构

UNIX 操作系统采用树型带勾连的目录结构，如下图所示。在这种结构中，一个文件的名字是由根目录到该文件的路径上的所有节点名按顺序构成的，相互之间用“/”分开。如文件 prog 的全路径名为：/usr/smith/prog，根目录用“/”表示。

根文件系统常用目录　根文件系统的常用目录举例如下表：

/bi 大部分可执行的UNIX命令和共用程序
/de 设备文件，如/dev/cd0
/et 系统管理命令和数据文件
/li C程序库
/us 存放用户的家目录和用户共用程序或文件
/t 临时工作目录，存放一些临时文件

家目录　UNIX 在创建用户名时，自动在/usr 目录下创建与用户名同名的子目录，如/usr/smith 子目录，这个子目录成为此用户的家目录（Home Directory）。家目录中有一个文件比较特殊：“.profile”(或“.login”)文件。当以该家目录的用户名登录 UNIX 时，会自动执行“.profile”文件。它有点类似于 DOS 的

AUTOEXEC.BAT 文件。“.profile”是 B/K shell 的启动文件，而“.login”是 C shell 的启动文件。“.profile”文件中常有一行：

PATH=$HOME:$HOME/bin

其中，PATH 类似 DOS 的 PATH,而$HOME 表示家目录。

1.2.2、能正确理解和掌握文件的权限以及文件类型

知道什么是文件的权限 即用户能对文件进行何种操作的权限

知道文件有那些权限

知道文件有那些类型 例如（设备文件、管道文件、普通文件）等

UNIX 重定向 将文件的标准输出重新定向输出到文件，或将数据文件作为另一程序的标准输入内容。如：

ls -l>file1 将ls -l命令显示的内容存到file1中，

ls>>file1 将ls 命令显示的内容附加存到file1的尾部

grep abc<file 将file 1 的内容作为grep abc命令的输入

1

其中，">"和">>"为输出重定向符，">"将输出内容存到重定向文件中，若文件存在，则先删除原有内容；">>"将输出内容存到重定向文件的尾部。

UNIX 管道 将一文件的输出作为另一文件的输入。如：

ls|more 将ls的输出作为more命令的输入

ps -ef|grep smith ps -ef的输出作为grep smith命令的输入

1.2.3、能正确掌握文件的基本操作

能对文件以及目录进行增加、修改、删除等。主要包括有 mv rm cp ls rmdir mkdir rename 等 UNIX 命令

UNIX 文件系统常用命令

	UNIX命令	UNIX命令举例	类似DOS命令
显示当前目录	pwd	pwd	cd
改变目录	cd	cd /usr	cd c:\usr
进入家目录	cd	cd	
创建目录	mkdir	mkdir abc	md abc
删除空目录	rmdir	rmdir abc	rd abc
删除目录及其内容	rm -r	rm -r abc	deltree abc
显示目录内容	ls	ls abc ls -l abc(文件长列表)	dir abc

		ls -a abc(所有类型文件) ls -d * (不进子目录)	
显示文本文件内容	cat	cat file1.c	type file1.c
一次一屏显示文本 文件内容	more	more file1.c	
拷贝文件	cp	cp file1 file2	copy file1 file2
移动(重命名)文件	mv	mv call.test call.list	move call.tst call.lst ren call.test bbb
删除文件	rm	rm call.list	del call.lst

1.3、 进程管理

1.3.1、 能正确了解 UNIX 系统下的进程的概念

了解什么是进程

程序的执行实例被称为进程（ process）

1.3.2、 能正确了解 UNIX 系统的进程调度

了解 UNIX 系统多进程的运行环境和模型

1.3.3、 能正确理解父子进程

知道什么是父进程、知道什么是子进程

应该是树行结构

子进程继承的父进程的属性:

实际 UID,GID 和有效 UID,GID.

. 环境变量.

. 附加 GID.

. 调用 exec()时的关闭标志.

. UID 设置模式比特位.

. GID 设置模式比特位.

. 进程组号.

. 会话 ID.

. 控制终端.

. 当前工作目录.

. 根目录.

. 文件创建掩码 UMASK.

. 文件长度限制 ULIMIT.

这些根据不同的调用，不同

常用的多进程编程的系统调用

1.fork()
功能:创建一个新的进程.

2.system()
功能:产生一个新的进程,　子进程执行指定的命令.

4.popen()
功能:初始化从/到一个进程的管道.

5.pclose()
功能:关闭到一个进程的管道.

6.wait()
功能:等待一个子进程返回并修改状态

7.waitpid()
功能:等待指定进程号的子进程的返回并修改状态

8.setpgrp()
功能:设置进程组号和会话号.

9.exit()
功能:终止进程.

10.signal()
功能:信号管理功能

11.kill()
功能:向一个或一组进程发送一个信号.

12.alarm()
功能:设置一个进程的超时时钟.

13.msgsnd()
功能:发送消息到指定的消息队列中.

14.msgrcv()
功能:从消息队列中取得指定类型的消息.

15.msgctl()
功能:消息控制操作

16.msgget()
功能:取得一个消息队列.

1.3.4、能正确掌握对进程的基本操作

知道 ps 命令的作用（以 SCO UNIX 系统为例）ps 命令：显示当前用户在系统中启动的进程。
/usr/bin/ps［选项］
-e：显示所有正在运行的进程。

-f：以完整的形式列出当前用户启动的进程。
-U：列出由这个用户启动的所有进程。
ps ：跟当前终端有关的进程；
ps -e
ps -f
ps -ef ：查看整个系统的进程

再如：ps -ef | grep inetd
结果：root 161 1 0 5 月 11 ? 0:09 /usr/sbin/inetd -s
zhangsw 5972 5860 0 18:33:59 pts/67 0:00 grep inetd

ps -ef | gerp 进程号
注意： 所有进程是树状结构：父 -----> 子。

知道 kill 命令的作用
知道 ps -ef 命令参数的输出中每一列的含义
知道如何查看指定用户有运行那些进程
知道如何杀掉指定进程号的进程

1.4、 常用指令

1.4.1、 VI 的操作

能使用 vi 进行简单的文本编辑，例如在指定位置插入一个字符，删除一个字符。删除一行、复制一行等操作

1.4.2、 FTP 的操作命令

能使用 FTP 命令在不同的系统中进行文件传输。并且能理解文本方式和二进制方式的区别

1.4.3、 其他操作

能掌握如下操作命令

tail	查看文件最后几行
wc	统计文件的字符数、行数、单词数
find	查找文件
grep	搜索文件中的指定字符串
more	分页显示
|	管道
man	帮助

diff　　文件比较

> >> < <<　　重定向命令

2、 程序部分

2.1、C 语言部分

2.1.1、 文件操作

要求掌握 fopen 函数、知道该函数打开文件/创建文件的方式有那些

知道什么是文件指针

知道如何对文件进行读操作，并能说明函数的参数及返回值的意义 要求掌握如下函数

（按字符读 fgetc 、读一行 fgets、 二进制方式读 fread）

知道如何对文件进行写操作，并能说明函数的参数及返回值的意义 要求掌握如下函数

(按字符写 fputc、 写一行 fputs、 二进制方式写 fwrite、 格式化写入 fprintf)

知道如何关闭文件 fclose 函数，并能说明为什么要关闭文件。

知道如何判断文件指针是否已经移到文件末尾

2.1.2、 字符串操作

知道什么是字符数组、字符串、字符串指针

知道函数 strcpy、strncpy、strcmp、strncmp、strcat、strstr、sprintf 的用法。

strncpy：字串复制

原型：char * strncpy(char *dest, char *src, size_t n);

功能：将字符串 src 中最多 n 个字符复制到字符数组 dest 中(它并不像 strcpy 一样遇到 NULL 就开始复制，而是等凑够 n 个字符才开始复制），返回指向 dest 的指针。

说明：

如果 n > dest 串长度，dest 栈空间溢出产生崩溃异常。

否则：

1）src 串长度<=dest 串长度,(这里的串长度包含串尾 NULL 字符)

如果 n=(0, src 串长度)，src 的前 n 个字符复制到 dest 中。但是由于没有 NULL 字符，所以直接访问 dest 串会发生栈溢出的异常情况。

如果 n = src 串长度，与 strcpy 一致。

如果 n = dest 串长度，[0,src 串长度]处存放 src 字串，(src 串长度，dest 串长度]处存放 NULL。

2）src 串长度>dest 串长度

如果 n =dest 串长度，则 dest 串没有 NULL 字符，会导致输出会有乱码。如果不考虑 src 串复制完整性，可以将 dest 最后一字符置为 NULL。

综上，一般情况下，使用 strncpy 时，建议将 n 置为 dest 串长度（除非你将多个 src 串都复制到 dest 数组，并且从 dest 尾部反向操作），复制完毕后，为保险起见，将 dest 串最后一字符置 NULL，避免发生在第 2)种情况下的输出乱码问题。当然喽，无论是 strcpy 还是 strncpy，保证 src 串长度<dest 串长度才是最重要的。

附：

Strcpy 和 Strncpy 的区别- -

第一种情况：

char* p="how are you ?";

char name[20]="ABCDEFGHIJKLMNOPQRS";

strcpy(name,p); //name 改变为"how are you ? "====>正确！

strncpy(name,p,sizeof(name)); //name 改变为"how are you ? " ====>正确！

strncpy(name,p, sizeof(name));//name 改变为"how are you ?" ======>正确！后续的字符将置为 NULL

第二种情况：

char* p="how are you ?";

char name[10];

strcpy(name,p); //目标串长度小于源串,错误！

name[sizeof(name)-1]='\0'; //和上一步组合，弥补结果，但是这种做法并不可取，因为上一步出错处理方式并不确定

strncpy(name,p,sizeof(name)); //源串长度大于指定拷贝的长度 sizeof(name)，注意在这种情况下不会自动在目标串后面加'\0'

name[sizeof(name)-1]='\0'; //和上一步组合，弥补结果

函数名: strncmp

功 能: 串比较

用 法: int strncmp(char *str1, char *str2, int maxlen);

说明:比较字符串 str1 和 str2 的大小，如果 str1 小于 str2，返回值就<0，反之如果 str1 大于 str2，返回值就>0，如果 str1 等于 str2，返回值就=0，maxlen 指的是 str1 与 str2 的比较的字符数。此函数功能即比较字符串 str1 和 str2 的前 maxlen 个字符。

函数名: strstr

功 能: 在串中查找指定字符串的第一次出现

用 法: char *strstr(char *str1, char *str2);

strstr 原型：extern char *strstr(char *haystack, char *needle);

用法：#include <string.h> (经过多次试验，这个头文件也可以不包含，并不影响该函数的使用。既然 MSDN 上要求， 可以不妨带上)

功能：从字符串 haystack 中寻找 needle 第一次出现的位置（不比较结束符 NULL)。

说明：返回指向第一次出现 needle 位置的指针，如果没找到则返回 NULL。

2.1.3、 内存操作

知道函数 malloc、free、memset、memcpy、memcmp 的用法

原型：extern void *memcpy(void *dest, void *src, unsigned int count);

用法：#include <string.h>

功能：由 src 所指内存区域复制 count 个字节到 dest 所指内存区域。

说明：src 和 dest 所指内存区域不能重叠，函数返回指向 dest 的指针。

注意：与 strcpy 相比，memcpy 并不是遇到'\0'就结束，而是一定会拷贝完 n 个字节。

原型：extern int memcmp(void *buf1, void *buf2, unsigned int count);

用法：#include <string.h>

功能：比较内存区域 buf1 和 buf2 的前 count 个字节。

说明：

当 buf1<buf2 时，返回值<0

当 buf1=buf2 时，返回值=0

当 buf1>buf2 时，返回值>0

知道为什么在程序中 采用 malloc 分配了内存在使用后一定要采用 free 去释放内存空间，知道如果不释放会有什么后果

内存分配方式有三种：

（1） 从静态存储区域分配。内存在程序编译的时候就已经分配好，这块内存在程序的整个运行期间都存在。例如全局变量，static变量。

（2） 在栈上创建。在执行函数时，函数内局部变量的存储单元都可以在栈上创建，函数执行结束时这些存储单元自动被释放。栈内存分配运算内置于处理器的指令集中，效率很高，但是分配的内存容量有限。

（3）从堆上分配，亦称动态内存分配。程序在运行的时候用malloc或new申请任意多少的内存，程序员自己负责在何时用free或delete释放内存。动态内存的生存期由我们决定，使用非常灵活，但问题也最多。

常见的内存错误及其对策如下：

内存分配未成功，却使用了它。

编程新手常犯这种错误，因为他们没有意识到内存分配会不成功。常用解决办法是，在使用内存之前检查指针是否为NULL。如果指针p是函数的参数，那么在函数的入口处用assert(p!=NULL)进行检查。如果是用malloc或new来申请内存，应该用if(p==NULL) 或if(p!=NULL)进行防错处理。

内存分配虽然成功，但是尚未初始化就引用它。

犯这种错误主要有两个起因：一是没有初始化的观念；二是误以为内存的缺省初值全为零，导致引用初值错误（例如数组）。

内存的缺省初值究竟是什么并没有统一的标准，尽管有些时候为零值，我们宁可信其无不可信其有。所以无论用何种方式创建数组，都别忘了赋初值，即便是赋零值也不可省略，不要嫌麻烦。

内存分配成功并且已经初始化，但操作越过了内存的边界。例如在使用数组时经常发生下标“多1”或者“少1”的操作。特别是在for循环语句中，循环次数很容易搞错，导致数组操作越界。

忘记了释放内存，造成内存泄露。

含有这种错误的函数每被调用一次就丢失一块内存。刚开始时系统的内存充足，你看不到错误。终有一次程序突然死掉，系统出现提示：内存耗尽。

动态内存的申请与释放必须配对，程序中malloc与free的使用次数一定要相同，否则肯定有错误（new/delete同理）。

释放了内存却继续使用它。

有三种情况：

（1）程序中的对象调用关系过于复杂，实在难以搞清楚某个对象究竟是否已经释放了内存，此时应该重新设计数据结构，从根本上解决对象管理的混乱局面。

（2）函数的return语句写错了，注意不要返回指向“栈内存”的“指针”或者“引用”，因为该内存在函数体结束时被自动销毁。

（3）使用free或delete释放了内存后，没有将指针设置为NULL。导致产生“野指针”。

别看free和delete的名字恶狠狠的（尤其是delete），它们只是把指针所指的内存给释放掉，但并没有把指针本身干掉。

用调试器跟踪示例7-5，发现指针p被free以后其地址仍然不变（非NULL），只是该地址对应的内存是垃圾，p成了“野指针”。如果此时不把p设置为NULL，会让人误以为p是个合法的指针。

如果程序比较长，我们有时记不住p所指的内存是否已经被释放，在继续使用p之前，通常会用语句if (p != NULL)进行防错处理。很遗憾，此时if语句起不到防错作用，因为即便p不是NULL指针，它也不指向合法的内存块。

```
char *p = (char *) malloc(100);
strcpy(p, “hello”);
free(p); // p 所指的内存被释放，但是p所指的地址仍然不变
…
if(p != NULL) // 没有起到防错作用
{
strcpy(p, “world”); // 出错
    }
```

动态内存会被自动释放吗？

函数体内的局部变量在函数结束时自动消亡。很多人误以为示例7-6是正确的。理由是p是局部的指针变量，它消亡的时候会让它所指的动态内存一起完蛋。这是错觉！

```
void Func(void)
{
```

```
char *p = (char *) malloc(100); // 动态内存会自动释放吗？
    }
```

我们发现指针有一些“似是而非”的特征：

（1）指针消亡了，并不表示它所指的内存会被自动释放。

（2）内存被释放了，并不表示指针会消亡或者成了NULL指针。

这表明释放内存并不是一件可以草率对待的事。也许有人不服气，一定要找出可以草率行事的理由：

如果程序终止了运行，一切指针都会消亡，动态内存会被操作系统回收。既然如此，在程序临终前，就可以不必释放内存、不必将指针设置为NULL了。终于可以偷懒而不会发生错误了吧？

想得美。如果别人把那段程序取出来用到其它地方怎么办？

2.1.4、 形参和值参

知道什么是形参全称为"形式参数"是在定义函数名和函数体的时候使用的参数,目的是用来接收调用该函数时传递的参数.

形参的作用是实现主调函数与被调函数之间的联系,通常将函数所处理的数据,影响函数功能的因素或者函数处理的结果作为形参.没有形参的函数在形参表的位置应该写void.main 函数也可以有形参和返回值,其形参也称为命令行参数,由操作系统在启动程序时初始化,其返回值传递给操作系统.

知道什么是值参只传递数值，在过程（函数）中对之所进行的改动，不会造成原始变量值的改变

2.1.5、 函数定义和返回值

知道如何定义函数、函数的返回值有什么作用。返回值可以将函数处理的数据提取给其他方法使用。

知道什么是函数声明函数声明是指函数的定义在后面，而前面需要对它进行调用，这样就需要预先作声明

知道什么是函数类型定义规定函数的返回值类型

2.1.6、 头文件的使用

理解 C 语言的头文件的作用和使用方法

1，头文件可以定义所用的函数列表，方便查阅你可以调用的函数；

2，头文件可以定义很多宏定义，就是一些全局静态变量的定义，在这样的情况下，只要修改头文件的内容，程序就可以做相应的修改，不用亲自跑到繁琐的代码内去搜索。

3，头文件只是声明，不占内存空间，要知道其执行过程，要看你头文件所申明的函数是在哪个.c文件里定义的，才知道。

4，他并不是 C 自带的，可以不用。

5，调用了头文件，就等于赋予了调用某些函数的权限，如果你要算一个数的 N 次方，就要调用 Pow（）函数，而这个函数是定义在 math.c 里面的，要用这个函数，就必需调用 math.h 这个头文件。

2.1.7、 Base 库的理解

能熟练掌握和使用 BASE 的函数

2.2、 数据库嵌入式程序部分

2.2.1、 能正确掌握编写嵌入式数据库程序

知道如何编写数据库嵌入式程序

知道如何对数据记录进行增加、删除、修改、插入操作

2.2.2、 能正确掌握索引

知道什么是索引

知道什么是唯一索引

知道为什么要用索引、索引有什么作用，尤其是对于唯一索引的正确使用

2.2.3、 能正确掌握数据库事物

知道什么是数据库事物

知道数据库事物有什么作用

知道什么情况下需要使用数据库事物

2.2.4、 能正确掌握数据库事物

知道如何使用数据库事物（打开数据库事物、提交数据库事物、回滚数据库事物）

2.2.5、 能正确掌握打开/关闭数据库的方法

2.2.6、 能正确掌握游标的类型与使用方法

知道什么是游标

知道游标有那些类型

知道在什么情况下需要使用游标以及使用注意事项

2.2.7、 能正确掌握数据库锁的概念

了解数据库锁的概念与作用，主要包括以下 3 个类型的锁(记录锁、页面锁、表级锁)

2.2.8、 能正确理解 SQLCODE、SQLERRD[2]

知道 SQLCODE 的作用

知道 SQLERRD[2]的作用

对于 INFORMIX 数据库，知道常用的 SQLCODE 的取值（操作成功、无记录、记录重复等）

3、 DBACCESS 操作要求

3.1、 基本操作

知道如何使用 dbaccess 对数据记录进行增加、删除、修改、插入的操作
对于 informix 要求能掌握如何对数据进行装载、卸载操作。例如（load unload）

3.2、 能正确使用 finderr 检查数据库错误原因

能使用命令 finderr 查看数据库错误原因

4、 其它

4.1、 能正确掌握工程文件 makefile 的使用

知道如何通过工程文件编译一个多文件组成的 C 语言程序

4.2、 能比较熟练的阅读计算机专业英语

能比较快速准确的阅读计算机专业英语